

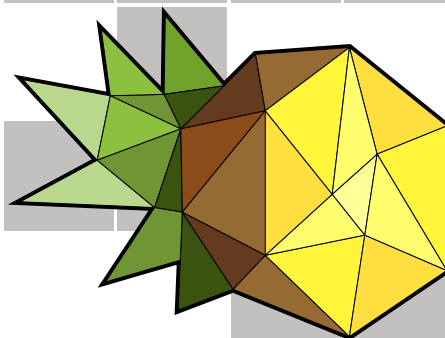
# AnAnaS : Analytical Analyzer of Symmetries

## User Guide

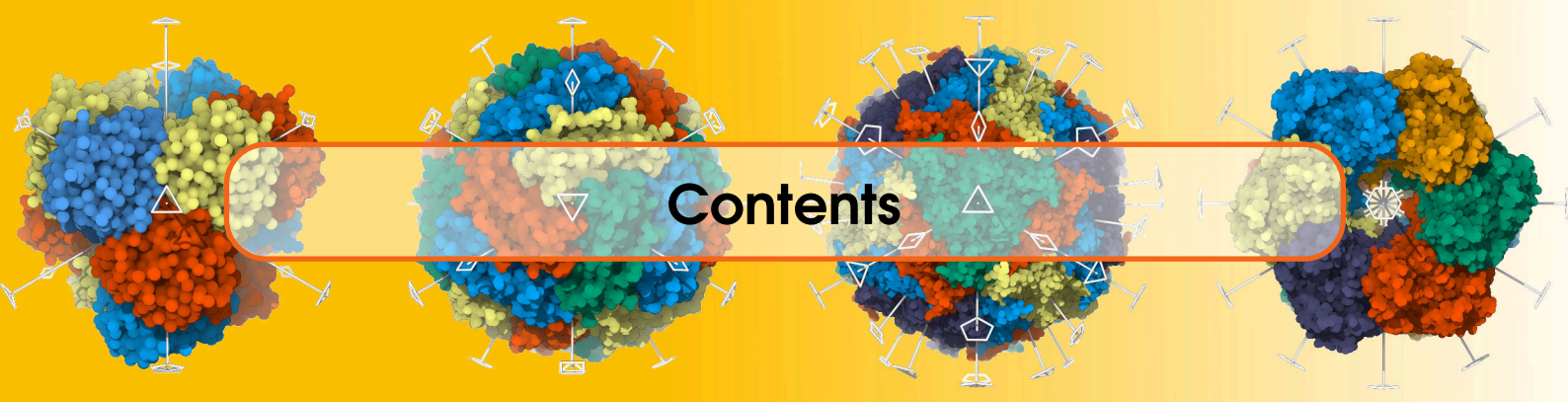
**Guillaume Pagès, Sergei Grudin**

Inria/CNRS Grenoble, France

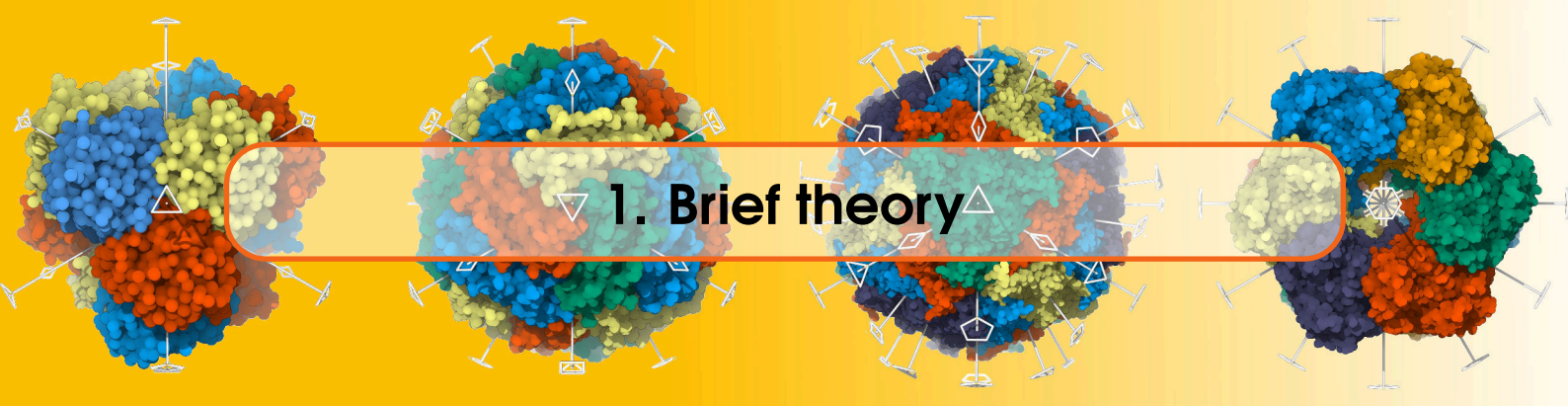
email : `sergei.grudin@inria.fr`



Copyright © 2017 Sergei Grudin / Guillaume Pagès



<b>1</b>	<b>Brief theory .....</b>	<b>3</b>
1.1	Molecular symmetry	3
1.2	Loss function	4
1.3	Workflow	4
<b>2</b>	<b>Program options .....</b>	<b>5</b>
2.1	Usage	5
2.2	Main options	6
2.3	Experimental options	7
<b>3</b>	<b>Examples .....</b>	<b>8</b>
3.1	Basic usage	8
3.2	Redirecting the result output	9
3.3	Printing out the permutations	9
3.4	Explicitly providing the permutations	10
3.5	Missing subunits	11
3.6	PyMol command	13
3.7	Output in the JSON format	13
<b>4</b>	<b>GUI interface .....</b>	<b>15</b>
	<b>References .....</b>	<b>16</b>
	<b>Index .....</b>	<b>17</b>



## 1.1 Molecular symmetry

All amino acids, except glycine, are chiral. Hence, symmetry groups that can be present in protein assemblies cannot contain any reflection, inversion, or improper rotation. The only remaining finite point groups are the cyclic ( $C_n$  for the cyclic group of order  $n$ ), dihedral ( $D_n$  for the dihedral group of order  $n$ ), tetrahedral  $T$ , octahedral  $O$  and icosahedral  $I$  groups. The last three ( $T$ ,  $O$  and  $I$ ) constitute the three cubic groups. It is convenient to introduce a symbol  $\Gamma \in \{C_n\}_{n>1} \cup \{D_n\}_{n>1} \cup \{T, O, I\}$  that will denote one of these point groups. Its cardinality, i.e. the number of its elements, will be denoted as  $|\Gamma|$ .

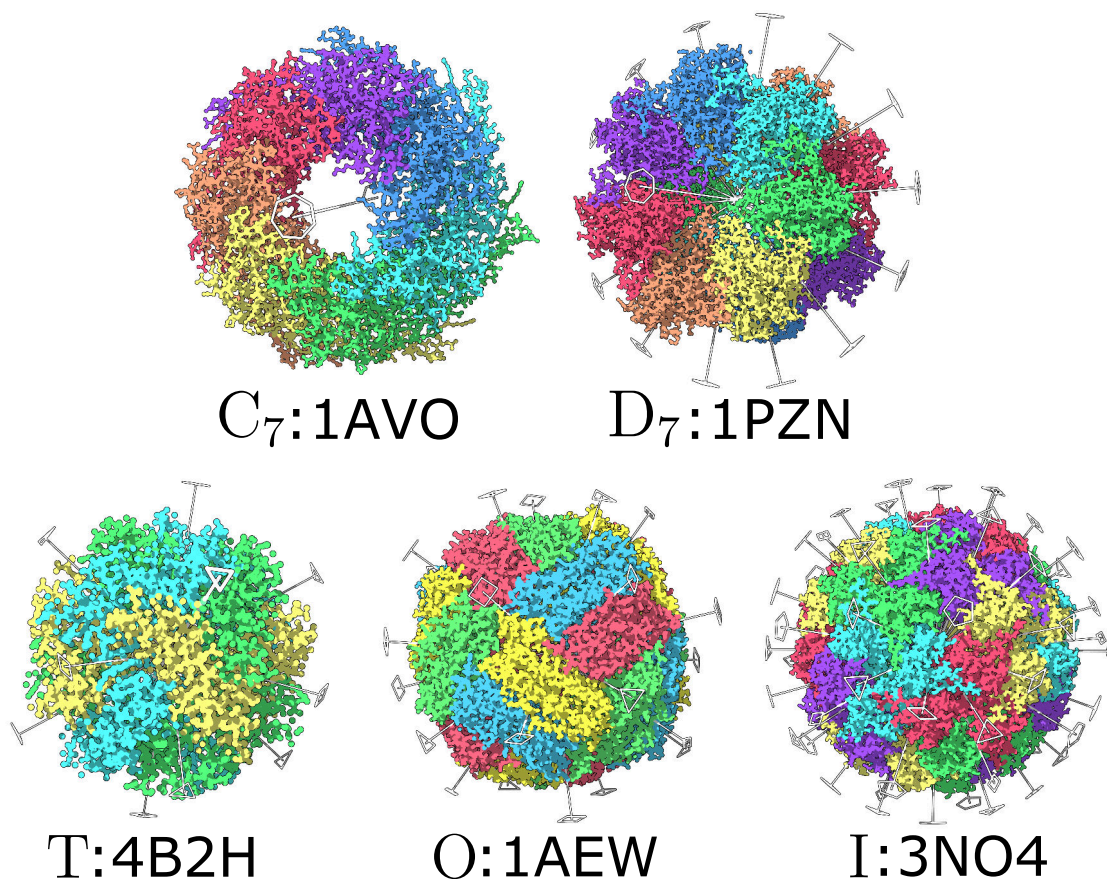


Figure 1.1: Five examples of symmetrical assemblies with their axes. These are a  $C_7$  cyclic assembly, a  $D_7$  dihedral assembly, a  $T$  tetrahedral assembly, an  $O$  octahedral assembly, and an  $I$  icosahedral assembly. The order  $n$  of each axis is represented with a regular  $n$ -gone, except of order 2 represented with a rhombus.

## 1.2 Loss function

Generally, point groups can have multiple rotation (or symmetry) axes of different order. Therefore, it is useful to formally define the loss function. First, let us associate each symmetry axis  $\vec{n}_k$  of order  $N_k$  with  $N_k - 1$  non-trivial rotations  $\hat{R}_k^i(2i\pi/N_k, \vec{n}_k)$  about this axis. Then, we define the loss function for a structure  $A$  as

$$\text{Loss}^2 = \sum_k \sum_{i=1}^{N_k-1} \text{RMSD}^2(A, \hat{R}_k^i(2i\pi/N_k, \vec{n}_k)A). \quad (1.1)$$

More generally, for a molecular point group  $\Gamma \in \{C_n\}_{n>1} \cup \{D_n\}_{n>1} \cup \{T, O, I\}$ , and an assembly  $A = \{\mathbf{a}_{i,j}\}_{N_s, N_a}$  consisting of  $N_s$  subunits, each composed of  $N_a$  atoms, we define the following loss function,

$$\text{Loss}^2 = \frac{1}{|\Gamma|N_sN_a} \min_{\sigma_g, r_g} \sum_{g \in \Gamma} \sum_{i=1}^{N_s} \sum_{j=1}^{N_a} \left( \mathbf{a}_{\sigma_g(i),j} - r_g(\mathbf{a}_{i,j}) \right)^2, \quad (1.2)$$

such that  $g \mapsto \sigma_g$  and  $g \mapsto r_g$  are bijective homomorphisms from  $\Gamma$  to subsets of  $\mathfrak{S}_n$  and  $SE(3)$ . The loss function is the sum of RMSDs between the original assembly and the rotated assemblies for every rotation in the group  $\Gamma$ . Here, we give the same weights to every atoms, but nothing prevents us from using weighted atoms, which would not change the algorithm. We should mention that this loss function is very natural, since it is only based on Euclidean 3D distances, no adjustable parameters are required and all the rotations  $r_g$  have equal importance. Without loss of generality we can assume that every subunit has the same number of reference points. Technically, we achieve it by performing a multiple sequence alignment of the subunits and keeping only the aligned parts for the subsequent analysis. More precisely, our reference points are located at the positions of the aligned  $C_\alpha$  atoms.

## 1.3 Workflow

Minimization of the loss function 1.2 requires optimization over the group of rotations, which is a *continuous optimization*, and over the group of permutations, which is a *discrete optimization*. In practice, we first apply a heuristic approach to determine the correspondences  $\sigma_g$  between the subunits, and then we analytically optimize the rotations. Overall, we solve the optimization problem in three steps,

1. Definition of asymmetric subunits
  2. Estimation of the permutations
  3. Optimization of the rotations
- (1.3)

The third step is an analytical continuous minimization. It gives the expected result with a machine precision. The first two steps are heuristics that assume the assembly to be symmetric enough, which allows to estimate the best correspondences between the subunits. This problem of estimating the best correspondence is discrete. More about the optimization problem and the continuous optimization can be found in the reference papers [1, 2, 3].



## 2. Program options

### 2.1 Usage

Typing the '--help' or '-h' flag produces the brief and more detailed description of the program options,

```
AnAnaS --help
```

```
*****
*-----AnAnaS : -----*
*-----Analytical Analyzer of Symmetries-----*
*-----Authors: Guillaume Pages & Sergei Grudinin-----*
*-----Reference 1: Analytical symmetry detection in cyclic protein assemblies.--*
*-----Reference 2: Analytical symmetry detection in dihedral-----*
*-----and cubic protein assemblies.-----*
*-----Copyright (c): Nano-D team, Inria/CNRS Grenoble, France -----*
*-----2014 - 2017.-----*
*-----e-mail: sergei.grudinin@inria.fr-----*
*-----*
*****
=====Parsing the Command Line=====
```

#### USAGE:

```
AnAnaS <input PDB> [--output <output file>] [-y] [-p] [-P <permutation
file>] [-d <dcd file>] [-j <json output>] [--logo] <symmetry
groups> ... [-h] [--version] [--log]
```

#### Where:

```
<input PDB>
  (required) PDB input file for the complex

--output <output file>
  Output path (by default, standard output will be used)

-y, --pymolOutput
  output pymol script to visualize the axes

-p, --permutationOutput
  Output the permutations

-P <permutation file>, --permutations <permutation file>
  Manually provide the permutation of the chains during symmetries

-d <dcd file>, --dcdfile <dcd file>
  Input trajectory in the dcd format

-j <json output>, --json <json output>
  File to output the results in json format

--logo
  prints logo and exits

<symmetry groups> (accepted multiple times)
```

```

symmetry groups

-h, --help
  Displays usage information and exits.

--version
  Displays version information and exits.

--log
  Displays ChangeLog information and exits.

```

Typing '--version' prints the current version of the program,

```
AnAnaS --version
```

```
AnAnaS version: 0.4, 21 November 2017
```

## 2.2 Main options

```
<input pdb> (required)
```

The first required argument is the path to an input PDB file, on which the symmetry detection will be performed.

```
--output <output file>
```

This options provides the output file name for the results. If nothing provided, the results will be written in the console.

```
-p, --permutationOutput
```

Enables the output of permutations used to compute the transformations, by default it's off.

```
-P <provide the permutation>, --permutations <provide the permutation>
```

This option allows the user to manually provide the permutation to be used to compute the transformation. This permutation WILL NOT be checked so it may return wrong results or even crash if this is wrong. For  $C_n$  symmetry, one permutation is expected corresponding to a rotation of  $\frac{2\pi}{n}$ . This permutation must also contain the non-present chains. For  $D_n$  symmetry, two permutations are expected, the first corresponding to a rotation of  $\frac{2\pi}{n}$ , the second corresponding to a rotation of  $\pi$ , with an axis orthogonal to the first one. For polyhedral symmetry, two permutations are expected, the first corresponding to a rotation of  $\frac{2\pi}{3}$ , the second corresponding to one of the 3 rotations of  $\pi$ , with the most collinear axis to the first one.

```
<symmetry groups> (accepted multiple times)
```

The symmetry groups to detect. Accepted groups are t, o, i, cn, dn with  $n$  being an integer bigger or equal to 2.

```
-y, --pymolOutput
```

Enables the output of the detected axes in a PyMol-compatible format, by default it's off.

```
-j <json output>, --json <json output>
```

Outputs the detected symmetry axes in the JSON format to the provided file.

```
-d <dcd file>, --dcdfile <dcd file>
```

Detects symmetry axes in each frame of the provided dcd trajectory file.

## 2.3 Experimental options

```
-r, --autoreplicate
```

Sets automatic replication of the protein, based on crystallographic information. Off by default.

```
-R <output file>, --autoreplicateOutput <output file>
```

This option provides a filename to output the result of the automatic replication. Only the alpha carbons of the replicas are present.

## 3. Examples

### 3.1 Basic usage

We will start by computing the symmetry of a tetrahedral assembly with pdb code 5x47.

**Example 1** To compute the group of symmetry for 5x47, type

```
AnAnaS 5x47.pdb
```

This will produce the following output in the terminal (we suppose you are starting the program from the terminal!),

```
=====Parsing the Command Line=====
No symmetry groups specified, will be detected automatically..... :
=====Reading PDB file=====
Read PDB file..... : 5x47.pdb
Number of chains read..... : 12
Number of atoms read..... : 13440
=====Cutting the input PDB file=====
=====Sequence alignment of chains=====
=====Auto-detecting Symmetry=====
=====Assumes that the assembly is complete=====
=====Detecting Symmetry=====
Symmetry group :..... : t
RMSD RMSD_R RMSD_T RMSD_Z RADGYR ORDER  AXIS X  AXIS Y  AXIS Z  CENTER X  CENTER Y  CENTER Z
0.769 0.447 0.515 0.355 29.397 3 0.536 -0.000 0.844 -26.543 -0.053 34.814
0.769 0.447 0.515 0.355 29.397 3 -0.536 0.000 -0.844 -26.543 -0.053 34.814
0.967 0.544 0.660 0.451 29.427 2 0.098 -0.777 0.621 -26.543 -0.053 34.814
0.873 0.539 0.556 0.403 29.482 3 0.423 0.897 0.127 -26.543 -0.053 34.814
0.945 0.511 0.537 0.587 29.467 3 -0.600 -0.198 0.775 -26.543 -0.053 34.814
0.833 0.551 0.457 0.427 29.469 2 0.153 -0.605 -0.781 -26.543 -0.053 34.814
0.945 0.511 0.537 0.587 29.467 3 0.600 0.198 -0.775 -26.543 -0.053 34.814
0.873 0.539 0.556 0.403 29.482 3 0.423 0.897 0.127 -26.543 -0.053 34.814
0.875 0.509 0.601 0.382 29.483 2 -0.983 -0.172 -0.060 -26.543 -0.053 34.814
0.928 0.531 0.542 0.534 29.492 3 0.713 -0.699 -0.058 -26.543 -0.053 34.814
0.928 0.531 0.542 0.534 29.492 3 -0.713 0.699 0.058 -26.543 -0.053 34.814
RMSD : 0.884844
Symmetry group :..... : d6
RMSD RMSD_R RMSD_T RMSD_Z RADGYR ORDER  AXIS X  AXIS Y  AXIS Z  CENTER X  CENTER Y  CENTER Z
36.302 27.749 15.867 17.206 29.436 6 -0.109 0.290 0.951 -26.543 -0.053 34.814
25.307 14.034 17.379 11.892 29.436 3 -0.109 0.290 0.951 -26.543 -0.053 34.814
37.159 14.402 32.336 11.302 29.459 2 0.145 -0.942 0.304 -26.543 -0.053 34.814
36.313 17.942 28.223 14.149 29.486 2 0.779 0.619 -0.099 -26.543 -0.053 34.814
25.307 14.034 17.379 11.892 29.436 3 -0.109 0.290 0.951 -26.543 -0.053 34.814
37.157 27.955 17.586 17.027 29.436 2 -0.109 0.290 0.951 -26.543 -0.053 34.814
36.302 27.749 15.867 17.206 29.436 6 -0.109 0.290 0.951 -26.543 -0.053 34.814
25.250 15.443 17.620 9.415 29.456 2 0.617 -0.730 0.293 -26.543 -0.053 34.814
0.878 0.518 0.598 0.379 29.483 2 0.983 0.171 0.061 -26.543 -0.053 34.814
36.288 17.964 28.244 14.015 29.468 2 0.924 -0.323 0.204 -26.543 -0.053 34.814
25.357 15.495 17.688 9.487 29.474 2 -0.366 -0.901 0.233 -26.543 -0.053 34.814
RMSD : 31.034982
Symmetry group :..... : c12
RMSD RMSD_R RMSD_T RMSD_Z RADGYR ORDER  AXIS X  AXIS Y  AXIS Z  CENTER X  CENTER Y  CENTER Z
45.385 29.592 30.303 16.304 29.459 12 -0.499 -0.586 0.638 -26.543 -0.053 34.814
45.385 29.592 30.303 16.304 29.459 12 -0.499 -0.586 0.638 -26.543 -0.053 34.814
45.991 34.236 27.145 14.357 29.459 6 -0.499 -0.586 0.638 -26.543 -0.053 34.814
37.610 25.119 23.981 14.438 29.459 12 -0.499 -0.586 0.638 -26.543 -0.053 34.814
```



```

47.950 34.982 27.971 17.117 29.459    3 -0.499 -0.586  0.638 -26.543 -0.053  34.814
45.783 33.252 26.950 16.251 29.459    4 -0.499 -0.586  0.638 -26.543 -0.053  34.814
37.610 25.119 23.981 14.438 29.459   12 -0.499 -0.586  0.638 -26.543 -0.053  34.814
45.783 33.252 26.950 16.251 29.459    4 -0.499 -0.586  0.638 -26.543 -0.053  34.814
47.950 34.982 27.971 17.117 29.459    3 -0.499 -0.586  0.638 -26.543 -0.053  34.814
45.991 34.236 27.145 14.357 29.459    6 -0.499 -0.586  0.638 -26.543 -0.053  34.814
35.649 19.970 27.126 11.674 29.459    2 -0.499 -0.586  0.638 -26.543 -0.053  34.814
RMSD : 43.942358
=====

```

The first section lists the group of symmetry provided. Here, we did not provide any, so, since the assembly has 12 chains, the program will try the tetrahedral  $T$ , the dihedral  $D_6$  and the cyclic  $C_{12}$  symmetry groups. The second section lists the structural parameters, the number of atoms and chains in the structure. The last section shows the results for each tested symmetry group. For each transformation it lists the associated RMSD, symmetry order, axis direction and axis position. The results show that the RMSD is much lower with the tetrahedral symmetry, indicating that this assembly has a tetrahedral symmetry.

### 3.2 Redirecting the result output

To redirect the result section of the output, use the `--output` option.

**Example 2** In this example, the result is redirected to the file `out.txt`.

```
AnAnaS 5x47.pdb t --output out.txt
```

### 3.3 Printing out the permutations

**Example 3** In this example we want to know which chain is compared with which rotated chain for each of the transformations.

```
AnAnaS 5x47.pdb t -p
```

The result part of the output will be:

```

=====Detecting Symmetry=====
RMSD RMSD_R RMSD_T RMSD_Z RADGYR ORDER  AXIS X  AXIS Y  AXIS Z  CENTER X  CENTER Y  CENTER Z
0.769 0.447 0.515 0.355 29.397    3  0.536 -0.000  0.844 -26.543 -0.053  34.814
(1, 2, 0, 10, 11, 3, 7, 8, 6, 4, 5, 9)
0.769 0.447 0.515 0.355 29.397    3 -0.536  0.000 -0.844 -26.543 -0.053  34.814
(2, 0, 1, 5, 9, 10, 8, 6, 7, 11, 3, 4)
0.967 0.544 0.660 0.451 29.427    2  0.098 -0.777  0.621 -26.543 -0.053  34.814
(3, 7, 11, 0, 10, 8, 9, 1, 5, 6, 4, 2)
0.873 0.539 0.556 0.403 29.482    3  0.423  0.897  0.127 -26.543 -0.053  34.814
(4, 5, 6, 7, 8, 9, 10, 11, 0, 1, 2, 3)
0.945 0.511 0.537 0.587 29.467    3 -0.600 -0.198  0.775 -26.543 -0.053  34.814
(5, 6, 4, 2, 3, 7, 11, 0, 10, 8, 9, 1)
0.833 0.551 0.457 0.427 29.469    2  0.153 -0.605 -0.781 -26.543 -0.053  34.814
(6, 4, 5, 9, 1, 2, 0, 10, 11, 3, 7, 8)
0.945 0.511 0.537 0.587 29.467    3  0.600  0.198 -0.775 -26.543 -0.053  34.814
(7, 11, 3, 4, 2, 0, 1, 5, 9, 10, 8, 6)
0.873 0.539 0.556 0.403 29.482    3  0.423  0.897  0.127 -26.543 -0.053  34.814
(8, 9, 10, 11, 0, 1, 2, 3, 4, 5, 6, 7)
0.875 0.509 0.601 0.382 29.483    2 -0.983 -0.172 -0.060 -26.543 -0.053  34.814
(9, 10, 8, 6, 7, 11, 3, 4, 2, 0, 1, 5)
0.928 0.531 0.542 0.534 29.492    3  0.713 -0.699 -0.058 -26.543 -0.053  34.814
(10, 8, 9, 1, 5, 6, 4, 2, 3, 7, 11, 0)
0.928 0.531 0.542 0.534 29.492    3 -0.713  0.699  0.058 -26.543 -0.053  34.814
(11, 3, 7, 8, 6, 4, 5, 9, 1, 2, 0, 10)
RMSD : 0.884844

```

In this example, the outputted permutations mean the following : the chain 0 (respectively 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), will be compared with the rotated chain 1 (respectively 2, 0, 10, 11, 3, 7, 8, 6, 4, 5, 9). It is important to note that the chains are labelled from 0, in their order of appearance in the pdb input file, regardless of their chain ID. If some chains are missing, they will still be represented in the permutation.

### 3.4 Explicitly providing the permutations

It might happen that the software is not able to correctly perceive the permutations corresponding to rotations. In this case, the user can provide a generating set of permutations. It is generally easy to guess if the permutations heuristic failed because it produces a ridiculously high RMSD measure.

**Example 4** In this example, 2ws6 has a pseudo  $D_3$  symmetry which makes the perception of a  $C_2$  structure difficult. The following command gives an RMSD of about 16.2 Å.

```
AnAnaS 2ws6.pdb c2 -p
```

The output will contain the following:

```
=====Detecting Symmetry=====
RMSD RMSD_R RMSD_T RMSD_Z RADGYR ORDER  AXIS X  AXIS Y  AXIS Z  CENTER X  CENTER Y  CENTER Z
16.197 15.614 2.989 3.100 14.127    2   0.661   0.529   0.533   14.673    0.206    1.411
(10, 3, 4, 1, 2, 11, 8, 9, 6, 7, 0, 5)
RMSD : 16.197311
```

**Example 5** But if we look for a  $D_3$  symmetry, the topology is well guessed correctly.

```
AnAnaS 2ws6.pdb d3 -p
```

The output will contain the following:

```
=====Detecting Symmetry=====
RMSD RMSD_R RMSD_T RMSD_Z RADGYR ORDER  AXIS X  AXIS Y  AXIS Z  CENTER X  CENTER Y  CENTER Z
1.169 0.692 0.692 0.639 14.126    2  -0.666  -0.533  -0.522   14.673    0.206    1.411
(2, 3, 0, 1, 10, 11, 8, 9, 6, 7, 4, 5)
1.156 0.651 0.800 0.522 15.024    3  -0.646   0.762   0.046   14.673    0.206    1.411
(4, 5, 6, 7, 8, 9, 10, 11, 0, 1, 2, 3)
0.909 0.521 0.462 0.584 14.160    2  -0.010   0.052  -0.999   14.673    0.206    1.411
(6, 7, 4, 5, 2, 3, 0, 1, 10, 11, 8, 9)
1.156 0.651 0.800 0.522 15.024    3  -0.646   0.762   0.046   14.673    0.206    1.411
(8, 9, 10, 11, 0, 1, 2, 3, 4, 5, 6, 7)
1.191 0.663 0.670 0.728 14.090    2  -0.656  -0.585   0.477   14.673    0.206    1.411
(10, 11, 8, 9, 6, 7, 4, 5, 2, 3, 0, 1)
RMSD : 1.120900
```

**Example 6** We can then use one of the permutations computed this way to get the  $C_2$  symmetry without other constraints.

```
AnAnaS 2ws6.pdb c2 -p -P "(2, 3, 0, 1, 10, 11, 8, 9, 6, 7, 4, 5)"
```

The output will contain the following:

```
=====Detecting Symmetry=====
RMSD RMSD_R RMSD_T RMSD_Z RADGYR ORDER  AXIS X  AXIS Y  AXIS Z  CENTER X  CENTER Y  CENTER Z
1.155 0.660 0.724 0.612 14.218    2   0.666   0.537   0.517   14.686    0.214    1.458
(2, 3, 0, 1, 10, 11, 8, 9, 6, 7, 4, 5)
RMSD : 1.155045
```

For dihedral and cubic symmetries, this option is much more tricky as two permutations are required.

The first one must correspond to a  $n$ -fold rotation for  $D_n$  or a 3-fold rotation for cubic symmetry groups, the second one must correspond to a 2-fold rotation. The two permutations also have to be generators of the entire group. They should be separated by a semi-colon (";").

We recommend to use this option very carefully in high-order symmetry assemblies. The example below shows the usage of multiple permutations.

**Example 7** Providing permutation for an octahedral assembly. Please note the file extension ("pdb1") of the input file 1kib.pdb1. It contains a biological assembly and has to be downloaded or constructed specifically.

```
AnAnaS 1kib.pdb1 o -P "(2, 22, 23, 5, 18, 6, 3, 20, 4, 19, 12, 17, 15,
1, 9, 10, 7, 21, 8, 14, 16, 11, 13, 0);(1, 0, 6, 7, 5, 4, 2, 3, 17, 16,
22, 23, 21, 20, 18, 19, 9, 8, 14, 15, 13, 12, 10, 11)"
```

The output will contain the following results:

```
=====Detecting Symmetry=====
  RMSD RMSD_R RMSD_T RMSD_Z RADGYR ORDER  AXIS X  AXIS Y  AXIS Z  CENTER X  CENTER Y  CENTER Z
0.192  0.112  0.112  0.108 33.386    2  -0.216 -0.573  0.790   -2.001   -2.001   -2.001
6.866  3.635  2.926  5.036 33.692    3  -0.740 -0.387  0.550   -2.001   -2.001   -2.001
6.866  3.624  2.963  5.023 34.296    4  -0.976  0.141 -0.165   -2.001   -2.001   -2.001
6.866  3.628  2.930  5.039 33.692    3  0.550 -0.740 -0.387   -2.001   -2.001   -2.001
6.866  4.132  2.878  4.668 35.182    2  0.807  0.591  0.017   -2.001   -2.001   -2.001
6.866  3.631  2.960  5.019 34.295    4  0.165  0.976 -0.141   -2.001   -2.001   -2.001
6.866  4.579  2.664  4.367 34.295    2  -0.165 -0.976  0.141   -2.001   -2.001   -2.001
0.020  0.015  0.009  0.009 36.047    3  -0.577 -0.578 -0.577   -2.001   -2.001   -2.001
0.191  0.111  0.112  0.107 33.386    2  -0.574  0.790 -0.216   -2.001   -2.001   -2.001
6.866  3.621  2.934  5.042 33.693    3  -0.387  0.550 -0.740   -2.001   -2.001   -2.001
6.866  3.617  2.966  5.026 34.297    4  -0.141  0.165  0.976   -2.001   -2.001   -2.001
6.866  4.557  2.678  4.382 34.297    2  0.141 -0.165 -0.976   -2.001   -2.001   -2.001
6.866  3.631  2.960  5.019 34.295    4  0.165  0.976 -0.141   -2.001   -2.001   -2.001
6.866  4.119  2.884  4.675 35.183    2  -0.017 -0.807 -0.590   -2.001   -2.001   -2.001
6.866  3.628  2.930  5.039 33.692    3  0.550 -0.740 -0.387   -2.001   -2.001   -2.001
0.020  0.015  0.009  0.009 36.047    3  0.577  0.578  0.577   -2.001   -2.001   -2.001
0.191  0.111  0.112  0.107 33.386    2  -0.790  0.216  0.574   -2.001   -2.001   -2.001
6.866  4.568  2.671  4.375 34.296    2  0.976 -0.141  0.165   -2.001   -2.001   -2.001
6.866  4.106  2.890  4.682 35.183    2  -0.591 -0.017 -0.807   -2.001   -2.001   -2.001
6.866  3.621  2.934  5.042 33.693    3  0.387 -0.550  0.740   -2.001   -2.001   -2.001
6.866  3.624  2.963  5.023 34.296    4  0.976 -0.141  0.165   -2.001   -2.001   -2.001
6.866  3.617  2.966  5.026 34.297    4  -0.141  0.165  0.976   -2.001   -2.001   -2.001
6.866  3.635  2.926  5.036 33.692    3  -0.740 -0.387  0.550   -2.001   -2.001   -2.001
RMSD : 6.074330
```

### 3.5 Missing subunits

The AnAnaS software supports missing subunits ONLY for cyclic symmetries. If there are some missing subunits, the result will give, for each permutation, the best rotation with the expected symmetry-constrained angle, but the different transformations will not form a group.

**Example 8** In this example, 2gza contains only 3 chains and is supposed to form a  $C_6$  assembly.

```
AnAnaS 2gza.pdb c6 -p
```

The symmetry detection output is:

```
=====Detecting Symmetry=====
  RMSD RADGYR ORDER  AXIS X  AXIS Y  AXIS Z  CENTER X  CENTER Y  CENTER Z
26.804 50.320    2  0.962  0.270  0.050   94.954  115.166  137.074
(1, 0, 3, 2, 5, 4)
```

3.278 60.904	3	0.995	0.094	-0.027	95.014	124.556	164.914
(2, 3, 4, 5, 0, 1)							
2.010 37.498	6	-0.985	-0.172	0.004	92.109	124.290	165.110
(3, 2, 5, 4, 1, 0)							
3.278 16.131	3	0.995	0.094	-0.027	93.768	124.439	164.948
(4, 5, 0, 1, 2, 3)							
2.010 16.143	6	-0.985	-0.172	0.004	91.484	124.181	165.112
(5, 4, 1, 0, 3, 2)							

After visual inspection, the 2gza structure seems fairly symmetric, there is no reason for one of the transformations to give an RMSD of 26 Å, we would expect an RMSD not bigger than 10 Å. After a closer look at the structure, we would expect the generator permutation to be circular "(1,2,3,4,5,0)" i.e. the chain 0 (respectively 1,2,3,4,5) will be compared with the chain 1 (respectively 2,3,4,5,0). However, none of the computed permutations in the above example perceived this circular order.

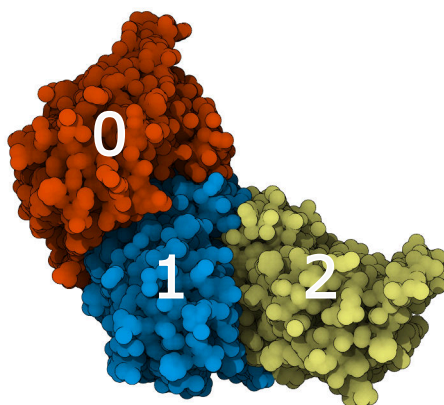


Figure 3.1: 2gza, a  $C_6$  assembly with 3 missing chains.

By explicitly providing the circular permutation, we obtain:

**Example 9** This is the same example as before, but here we provide the permutation explicitly.

```
AnAnaS 2gza.pdb c6 -P "(1,2,3,4,5,0)" -p
```

RMSD	RADGYR	ORDER	AXIS X	AXIS Y	AXIS Z	CENTER X	CENTER Y	CENTER Z
2.342	37.435	6	-0.994	-0.110	0.030	94.739	124.632	164.929
(1, 2, 3, 4, 5, 0)								
3.278	47.537	3	-0.995	-0.094	0.027	95.014	124.556	164.914
(2, 3, 4, 5, 0, 1)								
-nan(ind)	0.000	2	-1.000	-0.000	-0.000	0.000	0.000	0.000
(3, 4, 5, 0, 1, 2)								
3.278	23.429	3	-0.995	-0.094	0.027	93.768	124.439	164.948
(4, 5, 0, 1, 2, 3)								
2.342	23.384	6	-0.994	-0.110	0.030	93.742	124.522	164.959
(5, 0, 1, 2, 3, 4)								

This example shows that :

1. It is common that in the case of missing subunits the perceived permutation is wrong, it is a typical case where the user should provide the permutation manually.
2. Even if the chains 3, 4 and 5 are not present in the file, they have to be considered in the permutation.
3. In this example, there are 3 chains missing arranged in a way such that for one rotation (the one of order 2), none of the three chains match another chain. For this case, the RMSD and the best rotations are undefined.

### 3.6 PyMol command

We provide a PyMol script to visualize the predicted axes. Use the `-y` option to output the PyMol commands, then simply copy-paste it into your PyMol console. You can paste the full output, the non-relevant part will be ignored. Please note that the "cgo\_arrow.py" script has to be loaded in advance by 'run cgo\_arrow.py' in the PyMol console. The script can be found in the examples folder or at [http://pymolwiki.org/index.php/cgo\\_arrow](http://pymolwiki.org/index.php/cgo_arrow).

**Example 10** Getting the axes in a PyMol-compatible format for a tetrahedral assembly.

```
AnAnaS 5x47.pdb t -y
```

This commands outputs the following :

```
=====Detecting Symmetry=====
  RMSD RMSD_R RMSD_T RMSD_Z RADGYR ORDER  AXIS X  AXIS Y  AXIS Z  CENTER X  CENTER Y  CENTER Z
0.769 0.355 0.515 0.447 29.397 3 0.536 -0.000 0.844 -26.543 -0.053 34.814
cgo_arrow [-0.849513,-0.0646177,75.2899], [-52.2365,-0.0410524,-5.66119]
0.769 0.355 0.515 0.447 29.397 3 -0.536 0.000 -0.844 -26.543 -0.053 34.814
cgo_arrow [-53.84,-0.0403171,-8.18732], [0.754055,-0.0653531,77.816]
0.967 0.451 0.660 0.544 29.427 2 0.098 -0.777 0.621 -26.543 -0.053 34.814
cgo_arrow [-20.6028,-47.2009,72.5096], [-32.4832,47.0953,-2.88089]
0.873 0.403 0.556 0.539 29.482 3 0.423 0.897 0.127 -26.543 -0.053 34.814
cgo_arrow [-5.99524,43.5507,40.9694], [-47.0907,-43.6564,28.6592]
0.945 0.587 0.537 0.511 29.467 3 -0.600 -0.198 0.775 -26.543 -0.053 34.814
cgo_arrow [-56.1212,-9.84058,73.0655], [3.0352,9.73491,-3.43681]
0.833 0.427 0.457 0.551 29.469 2 0.153 -0.605 -0.781 -26.543 -0.053 34.814
cgo_arrow [-17.2959,-36.627,-12.3888], [-35.7901,36.5213,82.0175]
0.945 0.587 0.537 0.511 29.467 3 0.600 0.198 -0.775 -26.543 -0.053 34.814
cgo_arrow [1.97283,9.38336,-2.06294], [-55.0588,-9.48903,71.6916]
0.873 0.403 0.556 0.539 29.482 3 0.423 0.897 0.127 -26.543 -0.053 34.814
cgo_arrow [-5.99524,43.5507,40.9694], [-47.0907,-43.6564,28.6592]
0.875 0.382 0.601 0.509 29.483 2 -0.983 -0.172 -0.060 -26.543 -0.053 34.814
cgo_arrow [-85.963,-10.4222,31.2084], [32.877,10.3165,38.4203]
0.928 0.534 0.542 0.531 29.492 3 0.713 -0.699 -0.058 -26.543 -0.053 34.814
cgo_arrow [9.6233,-35.5348,31.8832], [-62.7093,35.4292,37.7455]
0.928 0.534 0.542 0.531 29.492 3 -0.713 0.699 0.058 -26.543 -0.053 34.814
cgo_arrow [-60.0063,32.7773,37.5264], [6.92033,-32.883,32.1023]
RMSD : 0.884844
```

You may then visualize the axes by typing the following command in the PyMol console:

```
run cgo_arrow.py          # needed to load the arrow script
cgo_arrow [-0.849513,-0.0646177,75.2899], [-52.2365,-0.0410524,-5.66119]
cgo_arrow [-53.84,-0.0403171,-8.18732], [0.754055,-0.0653531,77.816]
cgo_arrow [-20.6028,-47.2009,72.5096], [-32.4832,47.0953,-2.88089]
cgo_arrow [-5.99524,43.5507,40.9694], [-47.0907,-43.6564,28.6592]
cgo_arrow [-56.1212,-9.84058,73.0655], [3.0352,9.73491,-3.43681]
cgo_arrow [-17.2959,-36.627,-12.3888], [-35.7901,36.5213,82.0175]
cgo_arrow [1.97283,9.38336,-2.06294], [-55.0588,-9.48903,71.6916]
cgo_arrow [-5.99524,43.5507,40.9694], [-47.0907,-43.6564,28.6592]
cgo_arrow [-85.963,-10.4222,31.2084], [32.877,10.3165,38.4203]
cgo_arrow [9.6233,-35.5348,31.8832], [-62.7093,35.4292,37.7455]
cgo_arrow [-60.0063,32.7773,37.5264], [6.92033,-32.883,32.1023]
```

### 3.7 Output in the JSON format

To obtain the output in the JSON format, use the option `--json <jsonFilename>`. It will output a JSON array with the specification described in the `json-schema.json` file.

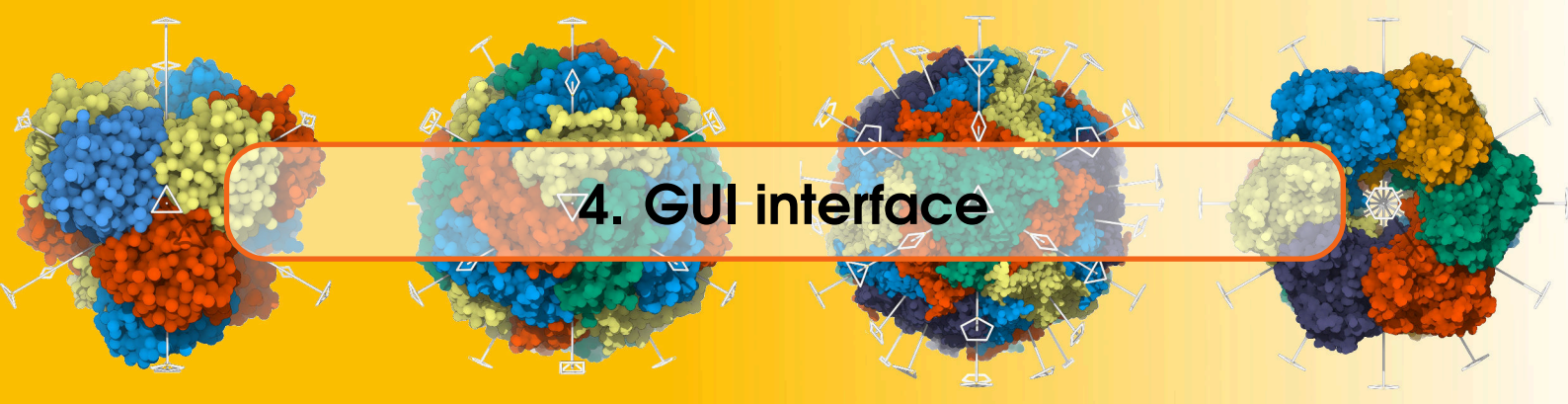


**Example 11** Getting the axes of the previous example in the JSON format.

```
AnAnaS 5x47.pdb t --json out.json
```

This example creates an output file out.json containing the following lines for each of the symmetry axes,

```
"transforms": [  
  {  
    "AXIS": [  
      0.535929763784527,  
      -0.000245769241457328,  
      0.84426253493054  
    ],  
    "CENTER": [  
      -26.5429868143223,  
      -0.0528350602150306,  
      34.8143356755033  
    ],  
    "ORDER": 3,  
    "P0": [  
      -0.849512978783817,  
      -0.0646176966753986,  
      75.2898584350204  
    ],  
    "P1": [  
      -52.2364606498608,  
      -0.0410524237546626,  
      -5.66118708401379  
    ],  
    "RAD_GYR": 29.3967753568363,  
    "RMSD": 0.769347896416487,  
    "RMSD_R": 0.355490833672876,  
    "RMSD_T": 0.515266893854406,  
    "RMSD_Z": 0.447238729307592  
  },  
]
```



A SAMSON module is available for the AnAnaS method at [samson-connect.net](http://samson-connect.net). It provides a convenient interactive graphical user interface. To use this module, select the structure on which you want to run the symmetry analysis. Then choose the symmetry group you want to test or keep it "Automatic" otherwise and click on "Compute Symmetry". The list of the results appears for every tested symmetry group. To visualize the axes, just click on an element in the list. You can also highlight specific axes by selecting them in the list, or move the camera along the chosen axis direction by double-clicking on the axis information in the list.

Figure 4.1 shows the SAMSON GUI module interface. The interface includes a 'Point Group Symmetry' dropdown menu set to 'Automatic', an 'Order' dropdown menu set to '2', and a 'Compute Symmetry' button. Below these controls is a table displaying the results of the symmetry analysis for various point groups.

Point group Symmetry	Axis O	RMSD	RMSD Radial	RMSD Tangential	RMSD Axial
▷ C2		55.580	17.064	49.176	19.486
▷ D2		0.894	0.421	0.579	0.535
▷ C3		35.927	15.117	23.969	22.083
▷ D3		36.080	15.104	28.319	16.483
▷ C4		44.724	15.418	28.576	30.755
▷ C6		39.070	16.698	26.863	22.935
▷ D6		32.873	14.089	20.276	21.703
▷ C12		45.385	16.304	30.303	29.592
▲ T		0.885	0.464	0.549	0.516
	3	0.769	0.355	0.515	0.447
	3	0.873	0.403	0.556	0.539
	3	0.945	0.587	0.537	0.511
	3	0.928	0.534	0.542	0.531
	2	0.967	0.451	0.660	0.544
	2	0.833	0.427	0.457	0.551
	2	0.875	0.382	0.601	0.509

The interface also includes a 'Clear' button at the bottom.

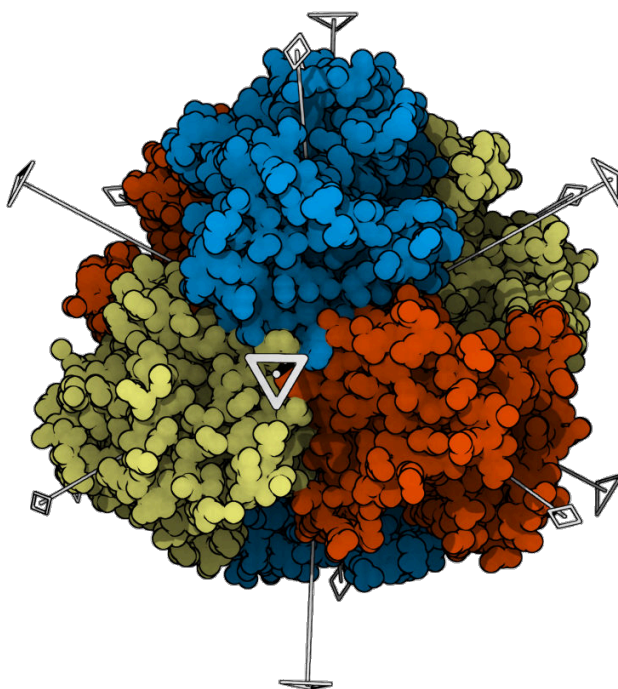
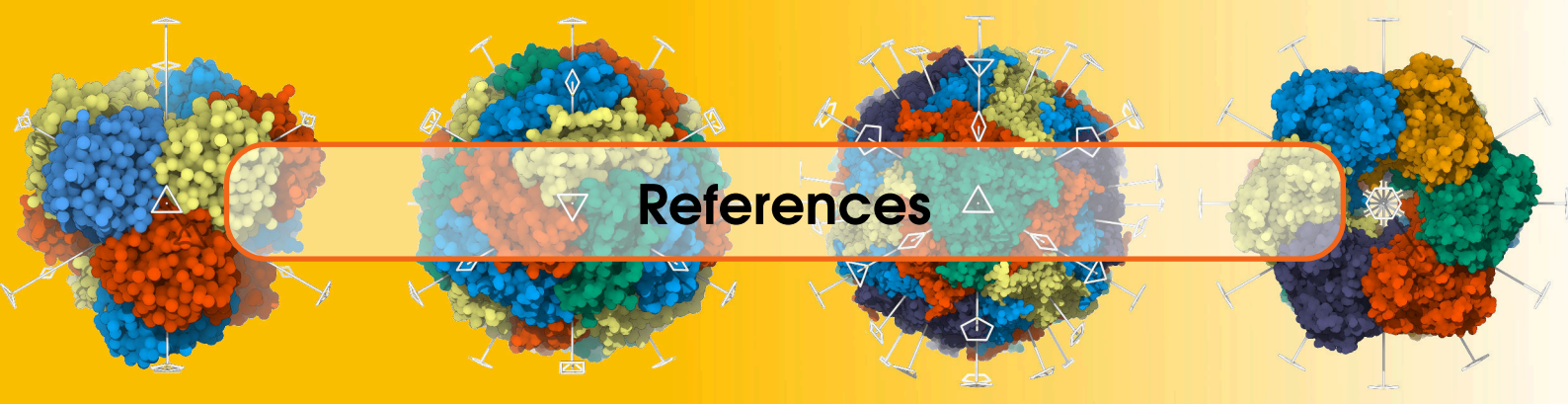
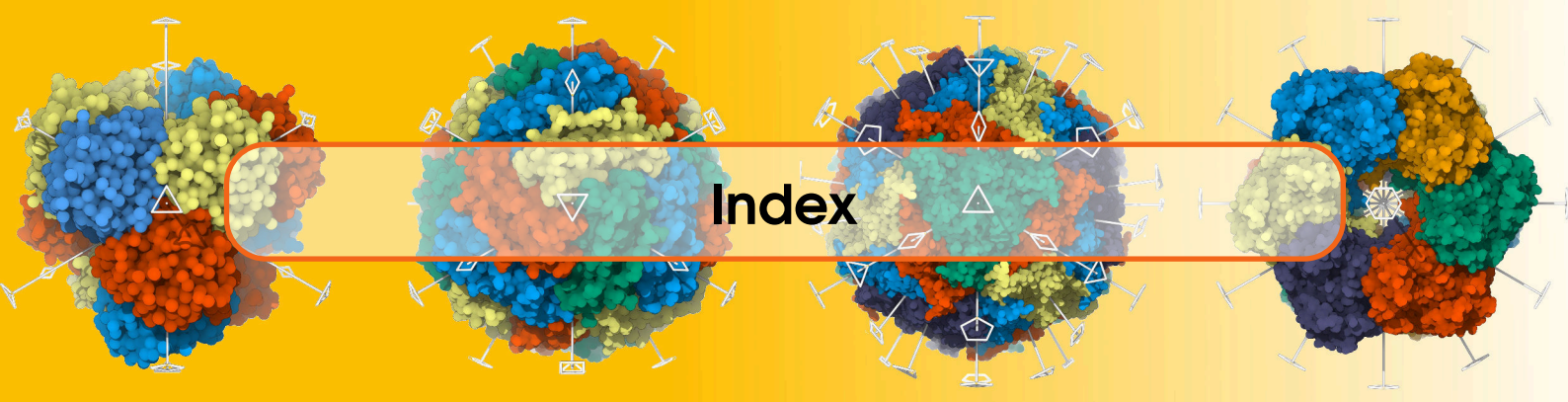


Figure 4.1: A SAMSON GUI module that provides an interactive graphical user interface for the AnAnaS method.



- [1] Petr Popov and Sergei Grudinin. “Rapid determination of RMSDs corresponding to macro-molecular rigid body motions”. In: *J Comput Chem* 35.12 (2014), pages 950–956 (cited on page 4).
- [2] Guillaume Pagès, Elvira Kinzina, and Sergei Grudinin. “Analytical symmetry detection in cyclic protein assemblies” (cited on page 4).
- [3] Guillaume Pagès and Sergei Grudinin. “Analytical symmetry detection in cubic protein assemblies” (cited on page 4).



# Index

## A

AnAnaS' Loss function .....	4
AnAnaS' Usage .....	5
AnAnaS' Workflow .....	4

## B

Basic examples .....	8
Brief theory .....	3

## E

Example with correctly perceived symmetry	10
Example with a cyclic group with missing sub-units .....	11
Example with a cyclic group with missing sub-units and an explicit permutation .	12
Example with a JSON output .....	14
Example with a PyMol-compatible output .	13
Example with a redirected output .....	9
Example with a tetrahedral symmetry group	8
Example with externally provided multiple permutations .....	11
Example with externally provided permutation .....	10
Example with incorrectly perceived symmetry .....	10
Example with the outputted permutations ...	9
Examples .....	8
Experimental options .....	7

## G

GUI interface .....	15
---------------------	----

## M

Main options .....	6
Missing subunits .....	11
Molecular symmetry .....	3

## O

Output in the JSON format .....	13
---------------------------------	----

## P

Printing out the permutations .....	9
Program options .....	5
Providing the permutations .....	10
PyMol command .....	13

## R

Redirecting the output .....	9
References .....	16

## S

SAMSON GUI illustration .....	15
-------------------------------	----